Accelerating Graph Analytics with oneAPI and Intel FPGAs

James Bickerstaff, Luke Kljucaric, Alan D. George
Department of Electrical and Computer Engineering, University of Pittsburgh
NSF Center for Space, High-performance, and Resilient Computing (SHREC)
Pittsburgh, PA, USA
{james.bickerstaff, luke.kljucaric, alan.george}@nsf-shrec.org

I. Introduction

Graphs are a popular and effective way to represent relationships between data points in a network due to their simple data abstraction and low storage cost per data point. However, as data continues to grow, the processing complexity of graph operations also significantly increases. Thus, there is a need to accelerate graph processing through specialized techniques and hardware in order to improve analytical throughput. This research uses the oneAPI toolkit with SYCL for FPGAs and leverages the increased productivity when creating designs, as compared to traditional hardware description language methodologies. The oneAPI toolkit is used in the creation of minimum-spanning-tree (MST) and breadth-first search (BFS) accelerators, evaluating the impact of different high-level design choices on the overall performance observed.

II. Approach

The overall architecture for both MST and BFS designs are very similar, and are based upon partitioning the graph between different kernels synthesized on the FPGA fabric. For the MST operation, Borůvka's algorithm is used, and all edges within the graph are evenly delegated to the kernels such that each identifies the cheapest edges to add to the overall tree concurrently. For BFS, the current "frontier," or list of vertices to explore, is split into partitions. The FPGA kernels then each process one partition at a time, traversing the frontier concurrently until no more remain.

One key focus of this research is investigating the impact of the different memory management techniques available within oneAPI. The two primarily observed are explicit Unified Shared Memory (USM), and USM with pipes incorporated. Pipes are FIFO structures used for transferring data between FPGA kernels. When using soley USM, a kernel will read all necessary data in, process it, and write the results out in that order. When incorporating pipes, however, three kernels are used in place of the single one. The three kernels aim to stream data from the host into kernel memory, process it, and stream the results out. In these studies, USM-only designs largely outperformed those using pipes.

Various numbers of kernels were tested to determine the optimal number of kernels for both MST and BFS designs.

This research was supported by SHREC industry and agency members and by the IUCRC Program of the National Science Foundation under Grant No. CNS-1738783.

Table I: Results

Operation	Vertices	Edges	FPGA	CPU
BFS	61M	403M	36 MTEPS	26 MTEPS
	129M	135M	74 MTEPS	25 MTEPS
	226M	240M	75 MTEPS	25 MTEPS
MST	25K	63K	11.3ms	16.8ms
	33K	78K	15.3ms	23.8ms
	49K	117K	23.5ms	35.6ms

FPGA and CPU columns are performace numbers related to the graph in the same row.

A single optimized kernel was first created before being replicated in the code base to synthesize many within the same design. A balance between high clock frequency, concurrent processing, resource utilization, and bandwidth estimations was found through this process of optimizing designs and benchmarking each result on different graphs. For MST, 12 kernels was found to be optimal, while 8 kernels was best for BFS.

An additional investigation done for the BFS operation was optimizing the size of partitions created. As the size increased, so did the area, clock frequency, and time required for each kernel to fetch data before starting computations. Thus, the granularity of partitions was fine-tuned such that data transfers were making full use of FPGA pipelines, while also not stalling other kernels from reading data. The MST operation holds 26,650 edges per kernel, while the BFS partitions each hold 8,000 edges max.

III. Evaluation

This research was conducted on the Intel DevCloud, with an Intel D5005 PAC equipped with a Stratix 10 SX as the FPGA accelerator. The host CPU and that used for baseline comparisons is the Intel Xeon Gold 6128. All graphs besides the (61M, 403M) one used in BFS benchmarking are real-world. The metric used for BFS performance is *Million Traversed Edges Per Second* as defined by the Graph 500 benchmark [1]. For MST, a measurement of runtime was used. The results displayed by our BFS design showcase speedups ranging from 1.4× to 3.0× over the baseline. For MST, our results showcase a speedup of ∼1.5× across all graphs tested.

References

[1] "Benchmark specification - graph 500," Nov 2021. [Online]. Available: https://graph500.org/?page_id=12